manuale

5

Per i possessori di computer

ZX SPECTRUM e TI 99/4A

LINGUAGGIO MACCHINA: 1ª LEZIONE

PSEUDOCODICE:
I numeri a caso
La smazzata

IL BASIC DELLO SPECTRUM
3ª lezione

EV EDITORIALE VIDEO

COMPUTING VIDEOTECA numero 6. Il manuale è sempre più ricco, sempre più indispensabile per coloro che possiedono ZX SPECTRUM oppure TI 99/4A e intendono utilizzare al meglio le capacità della macchina, oltre che migliorare le proprie cognizioni di informatica.
In questo numero c'è la nuova lezione di Pseudocodice che presenta un'esercitazione pratica molto interessante per coloro che si sono accostati alla programmazione: la distribuzione di un mazzo di 52 carte a quattro giocatori. La cosiddetta "smazzata" è indispensabile nel bagaglio pratico di un programmatore.
E arriviamo subito al piatto forte di questo numero: la prima lezione di LINGUAGGIO MACCHINA. L'autore ha voluto porre in rilievo un concetto che spesso viene preso sotto gamba: per diventare esperti di L/M è utile e necessario puntualizzare i concetti di elettronica digitale, ed è giusto questo l'argomento della prima puntata.
Prosegue frattanto il corso di Basic dello Spectrum che, in questo numero, si occupa di parecchie funzioni principali che di certo sono già familiari alla maggior parte dei lettori. Tuttavia è utile inquadrarle in un discorso scientifico che integra perfettamente la pratica.
La cassetta presenta dieci programmi (5 per lo Spectrum e 5 per il TI 99/4A) che vanno ad arricchire la nastroteca dei lettori affezionati.
Per lo Spectrum presentiamo Square (gioco di abilità), Cavern (una magnifica battaglia spaziale), 3D Graff (programma che fornisce perfetti grafici tridimensionali), Ping-pong (simula a meraviglia una partita con la racchetta piccola) e Space Traffic (divertente gioco spaziale di abilità).
Il lato TI 99/4A, non meno ricco e piacevole, presenta: Cassaforte (mette alla prova la vostra abilità: avete 9 tentativi per scoprire la combinazione!), Media-voti (utilissimo per gli studenti), Lettere assassine (divertente gioco d'azione), Labirinto (gioco classico ma appassionante), Help me (devi salvare una principessa lottando contro vari nemici).
Per coloro che abbiano perso qualche numero, lo ripetiamo ancora, gli arretrati si ottengono inviandone richiesta su vaglia intestato a Editoriale VIDEO - Via Castelvetro 9 - 20154 Milano. La richiesta si può anche inviare per lettera provvedendo al pagamento con assegno. Il prezzo di ciascun numero arretrato è di L. 15.000, comprensivo di tutte le spese di invio.
Arrivederci al prossimo numero, fra un mese, che presenterà, fra l'altro, la nuova lezione del corso di Linguaggio Macchina.

Editoriale VIDEO - Direttore: Antonio Lucarella - Coordinamento tecnico: Roberto Treppiedi - Hanno collaborato: Franco Longoni, Marco Affer, Alberto Barbati, Massimo Cellini, Fabrizio Iotti, Maria Russino, Aldo Campanozzi, Roberto Muraglia, Carlo Mantegazza, Stefano Milanesi, Maurizio Monteverdi, Fabio Macchioni, Tito Caponio, Sebastiano Pastore, Domenico Cellamare, Roberta Di Pietro, Alessandro Vallone, Fabio Rossi, Arnaldo Restelli, Dino Ticli, Igino Zaffaina, Giovanna Zampella - Stampa: La New Graf Milano - Fotocomposizione: ERREGI Milano - reg. del Trib. di Milano n. 519 del 10/11/84 - Stampato in Italia

• NUMERI A CASO
• LA SMAZZATA

INTRODUZIONE

Eccoci giunti al 6° articolo della serie. Abbiamo ormai un bagaglio di conoscenze adeguate ad affrontare un problema complesso e a progettarne la risoluzione.
Il problema che ci proponiamo di presentare questo mese è LA DISTRIBUZIONE DI UN MAZZO DI 52 CARTE DA GIOCO A 4 giocatori. Nonostante l'apparente semplicità del problema vedremo che il programma sarà piuttosto ricco di spunti. Considerate anche che il tempo di esecuzione su un P.C. è di oltre 40 secondi: ciò significa che il problema è davvero poderoso dal punto di vista del calcolo.
Inoltre sotto la superficie del problema si nasconde un importante tema: la CASUALITÀ degli eventi coinvolti.
È proprio questa CASUALITÀ che farà da linea guida per tutto questo articolo, che come al solito, non si propone solo di presentare delle soluzioni, ma anche di suggerire delle idee di progetto ai nostri lettori.
Infine il programma BASIC che accompagna questo articolo, può essere utilizzato come punto di partenza per la più ambiziosa realizzazione di un giocatore "automatico" per qualche gioco di carte.

LA CASUALITÀ

Un programma è un insieme di istruzioni chiare e non ambigue. Lo pseudocodice insegna che la logica del programma è ben rappresentata dalle figure fondamentali e che i predicati di controllo funzionano come cerberi inflessibili.
Come possiamo allora rappresentare la CASUALITÀ propria di certi eventi naturali all'interno di un programma?
Ricordiamo che la CASUALITÀ di un evento è la proprietà di un evento di essere incerto: cioè di poter manifestarsi in un modo oppure in un altro del tutto diverso. Oppure addirittura di non manifestarsi proprio.
Un esempio tipico è la scelta di una carta in un mazzo di 52 carte. Definiamo EVENTO in questo esempio il fatto che dal mazzo venga estratta una certa carta. In una SMAZZATA (distribuzione di 52 carte a 4 giocatori) avvengono 52 di questi eventi.
Proviamo a pensare come potremmo programmare una simulazione di smazzata attraverso un programma in pseudocodice. Se perdete 5 minuti di tempo per pensarci vi renderete conto che è IMPOSSIBILE scrivere un programma che realizzi una scelta VERAMENTE a caso tra 52 carte, usando solo istru-

zioni DETERMINISTICHE, come quelle studiate fino ad ora, senza che noi programmatori siamo in grado di sapere ESATTAMENTE quale scelta viene fatta dal nostro programma. E questo distrugge la CASUALITÀ dell'evento.
Una proposta di soluzione potrebbe essere la seguente:

INIZIO. Scelta a caso tra 52 carte.
"Acquisici da tastiera il numero N"
"Scegli la N-sima carta del mazzo"
FINE.

Con l'istruzione di INPUT del numero N noi lasciamo la scelta della carta alla persona seduta davanti al PC.
Allora succede che chi si trova a immettere il numero si assume la responsabilità di scegliere a caso, e perciò si RINUNCIA a PROGRAMMARE la casualità, delegando una persona a scegliere secondo il caso.
Noi però non saremo mai soddisfatti, perché sapremo sempre che carta sceglierà per ogni N (l'N-sima è sempre la medesima).
Come uscire da questo circolo vizioso?
La soluzione esiste e consiste nel mettersi in una prospettiva diversa. In realtà la CASUALITÀ esiste solo perché noi non siamo in grado di **prevedere** con certezza se un certo evento si manifesta o no. In pratica abbiamo l'**impressione** della casualità quando non **sappiamo** prevedere quale risultato seguirà ad una azione.
Nel caso del mazzo di cui si sceglie una carta, assumere questo atteggiamento è come dire: se **sapessimo** esattamente la disposizione di tutte le carte nel mazzo dopo averle mischiate, la nostra scelta non sarebbe più CASUALE ma cosciente e voluta, così come nel caso del programma per cui scegliendo "a caso" un numero, in pratica **sapevamo** la carta selezionata.
Ecco allora la strada per risolvere l'arcano: **Si scrive un programma che generi un numero che non possiamo neanche lontanamente prevedere**, e questo numero sarà usato per scegliere una carta.
Sono sicuro che molti di voi si sentiranno presi in giro: "Ma come, - diranno, - la casualità dell'evento appartiene all'evento, e non dipende dal fatto che io sappia o no se esso capita: l'incertezza è nell'evento, non in me stesso!!"
Costoro possono essere orgogliosi di avere questa opinione filosofica, perché sono in ottima compagnia: Einstein stesso la pensava così ma solo... "in linea di principio".
Nella pratica è dimostrabile matematicamente che questa posizione e quella descritta sopra sono equivalenti nei loro effetti: sia che l'incertezza stia nell'evento, o in chi osserva l'evento, non ci sono differenze pratiche.
In uno o nell'altro dei due atteggiamenti "filosofici" le cose re-

stano nello stesso modo. Naturalmente noi scegliamo il secondo atteggiamento perché è l'unico programmabile.
La comunità dei programmatori battezza perciò CASUALE qualunque programma che generi dei numeri senza che si sappia prevedere (con certezza) quale successivo numero il programma genererà.
"Ma come" - dirà qualcuno - "neanche chi ha scritto il programma sa che numero estrarrà il programma stesso?" La risposta è NO.
Analizzando uno di questi programmi estrattori di numeri A CASO (in inglese RANDOM), si vede che però una certa legge di generazione esiste. Solo che essa è così bizzarra e complessa che nessun essere umano in modo da avere una percezione "istintiva" di una legge sottostante. Il programmatore che vede il programma estrarre un numero lo vede agire totalmene A CASO.

I NUMERI CASUALI (RANDOM) IN BASIC

In BASIC con una istruzione sola (che richiama un programma già scritto) si genera un numero con la virgola compreso tra 0 e 1.
Per esempio scrivendo:

```
10 A = RND
20 PRINT A
```

e lanciando il programma più volte si ottengono numeri tipo 0.538 oppure 0.997, insomma qualunque numero tra 0 e 1 compresi.
Per ottenere un numero in un intervallo tra, supponiamo, 0 e 50, basta moltiplicare il numero prodotto da RND per 50.
Per ottenere un numero tra 1 e 52 dovremo allora scrivere:

```
A = (RND* 52) + 1.
```

Se vogliamo un numero INTERO (cioè senza virgola) tra 1 e 52, scriveremo:

```
A = INT (RND*52) + 1.
```

Il nostro problema della scelta della carta può essere ora risolto in questo modo:

```
INIZIO. Scelta di 1 carta su 52
  "Assegna ad N un numero
  random tra 1 e 52"
  "Scegli la carta N-sima"
FINE.
```

Lanciando questo programma, il programmatore non potrà prevedere quale carta il programma estrarrà, esattamente come se incaricasse un amico di scegliere a caso un numero.

LA SMAZZATA

Il problema di distribuire 52 carte a 4 giocatori non è una semplice ripetizione di 52 scelte di una carta. C'è una notevole variante di cui tenere conto: dopo ogni estrazione bisogna trovare il modo di "eliminarla dal mazzo": essa non deve più essere estraibile.
Poi notiamo che vogliamo un sistema per rappresentare le 52 carte del mazzo, in modo da

non dover fare 52 scomode assegnazioni.
Pensiamo di rappresentare una carta mediante 2 variabili con indice: SEME(I) che rappresenta il seme della I-esima carta, e VALOR(I) che ne rappresenta il valore.
Per usare variabili numeriche (in modo da gestire i valori in modo numerico) usiamo i numeri 1,2,3 e 4 per rappresentare Picche, Cuori, Quadri e Fiori rispettivamente.
Analogamente i numeri 13,12 e 11 rappresentano Re, Donna e Asso rispettivamente.
A questo punto il caricamento (la preparazione) del mazzo ordinato si può fare così:

```
INIZIO. Caricamento del mazzo
 RIPETI
 PER I DA 1 A 4
  RIPETI
  PER J DA 1 A 13
   K=(I-1)*13+J
   SEME (K)=I
   VALOR (K)=J
  FINE
 FINE
FINE.
```

Questo programma inserisce in SEME(K) e VALOR(K) tutte le carte ordinate da Picche a Fiori. La gestione di K è divertente e lascio al lettore il compito di capirne la logica (suggerimento: in questo modo K assume i valori da 1 a 52 in sequenza).
Dopo aver creato il mazzo pensiamo alla distribuzione:

```
INIZIO. Distribuzione
 RIPETI
 PER I DA 1 A 13
  RIPETI
  PER J DA 1 A 4
   "Scegli a caso una carta
   tra quelle rimaste"
   "Dai la carta al giocatore
   J"
   "Elimina la carta estratta
   dal mazzo.
  FINE_RIPETI
 FINE_RIPETI
FINE.
```

(Osservate la scelta del giocare J-simo, in quanto J oscilla tra 1 e 4).
Per ogni giocatore dobbiamo pensare a come memorizzare che ha ricevuto la carta estratta.

Pensiamo a una variabile a doppio indice: il primo indica il giocatore, il secondo il numero progressivo della sequenza di carte a lui distribuita.
Però per ogni carta noi vogliamo sapere sia il valore che il seme. Perciò avremo una coppia di variabili a 2 indici: chiamiamole GSEM (H,L) e GVAL (H,L).
Per esempio se GSEM (4,10) = 2 e GVAL (4,10) = 1 allora il giocatore 4 ha come 10-ma carta in mano un asso di Quadri. Chiaro? Un altro esempio: GSEM (3,2) = 1 e GVAL (3,2) = 11 significa che la 2ª carta del 3° giocatore è un Fante di Picche.
Ecco come si riscrive il nostro programma, con queste ipotesi:

```
INIZIO. Distribuzione.
 RIPETI
 PER I DA 1 A 13
  RIPETI
  PER J DA 1 A 4
   "Scegli una carta a caso:
```

```
    la carta di indice X"
    GSEM (J,I) = SEME(X)
    GVAL (J,I) = VALOR(X)
    "Elimina la carta dal
    mazzo"
   FINE_RIPETI
  FINE_RIPETI
 FINE.
```

Osservate il gioco di J ed I, che contano i giocatori ed i giri rispettivamente.

Siamo arrivati al problema cruciale: come scegliere nel mazzo una carta tra 52 al primo giro una tra 51 al secondo ... fino ad 1 tra 2 al penultimo giro?

Bisogna introdurre una variabile P al posto del 52 nell'istruzione descritta al paragrafo precedente:

```
X = INT(RND*(P)) + 1
```

e poi far decrescere P da 52 a 1 levando 1 ad ogni giro.

L'ultimo problema rimasto è: come eliminare dal mazzo una carta già scelta? Ecco un'idea. Una volta scelta la carta X ed assegnati i valori al giocatore J, giro I, facciamo un riordinamento dei vettori SEME(N) e VALOR(N). Spostiamo tutto il mazzo in alto di una carta di un posto a partire dalla carta X + 1. In questo modo, alla prossima scelta tra 1 e (P-GIRO) cadremo sempre su una carta non ancora scelta.

Meglio ancora se mettiamo Ø sia in seme che in valore nell'ultima carta del mazzo. Così pian piano i vettori si riempiono di zeri dal fondo, indicando le carte eliminate.

Ecco lo pseudocodice:

```
INIZIO_ Elimina carta X dal
mazzo.
 RIPETI
 PER N DA 1 A 52
   VALOR(N) = VALOR(N + 1)
   SEME(N) = SEME(N + 1)
 FINE_RIPETI
FINE.
```

(Per usare con eleganza questo trucco, conviene mettere una 53 carta fittizia nel mazzo il cui valore e seme valgono Ø).

Con questo i pezzi del nostro programma sono tutti a posto. Basta creare il programma di stampa delle carte dei giocatori al termine della smazzata.

Per fare questo usiamo un vettore di decodifica: per ogni numero di VALOR assegnamo un carattere ASCII che ne rappresenta la carta. Lo stesso per il SEME.

Chiamiamo il primo vettore RV$, ed il secondo RS$.

Allora possiamo anche pensare di raggruppare le carte per seme, in output. Otteniamo:

```
INIZIO Stampa.
 RIPETI
 PER I DA 1 A 4
   Stampa "Giocatore" I
     Stampa seme: RS$(J)
   PER K DA 1 A 13
     SE GSEM (I,K) = J
     ALLORA Stampa RV$
     (GVAL(I,K))
     FINE_SE
   FINE RIPETI
 FINE_RIPETI
FINE.
```

Ecco fatto, il programma è progettato. Chi vuole può divertirsi a realizzarlo per conto proprio.

Per chi si è già...stufato il programma BASIC è già pronto in figura 1.
Memorizzatelo e fatelo girare. Il risultato è simile a quello mostrato in figura 2, ma leggete attentamente la didascalia.
L'istruzione 131 apparentemente non ha molto senso. Ma provate a levarla...avrete l'emozione di attendere con "suspense" che il programma termini 40 secondi (un'eternità) di elaborazione.
Arrivederci

M.S.

```
10 DIM SEME(53),VALOR(53),GVAL(4,13),GSEM(4,13),RS$(4),RV$(13)
15 RS$(1)="P":RS$(2)="C":RS$(3)="Q":RS$(4)="F"
16 RV$(1)="1":RV$(2)="2":RV$(3)="3":RV$(4)="4"
17 RV$(5)="5":RV$(6)="6":RV$(7)="7":RV$(8)="8"
18 RV$(9)="9":RV$(10)="10":RV$(11)="J":RV$(12)="Q":RV$(13)="K"
20 REM Smazzata
30 GOSUB 2000  'Richiama Carica-mazzo
40 P=52          'numero di carte rimanenti nel mazzo
50 FOR I=1 TO 13
60      FOR J=1 TO 4
80          X=INT(RND*(P))+1
90          GSEM(J,I)=SEME(X)
100         GVAL(J,I)=VALOR(X)
110         GOSUB 3000    ' richiama elimina carta
120         P=P-1
130     NEXT J
131   PRINT "GIRO NUMERO ..." I
140 NEXT I
150 ' Stampa le carte smazzate
160 FOR I=1 TO 4
170   PRINT " GIOCATORE Numero " I
180   FOR J=1  TO 4
190     PRINT RS$(J); "  :";
200     FOR K=1 TO 13
210       IF GSEM(I,K)=J THEN PRINT RV$(GVAL(I,K));" ";
220     NEXT K
225     PRINT
230   NEXT J
240 NEXT I
250  STOP
2000  ' carica il mazzo di carte
2010  FOR I=1 TO 4
2020    FOR J=1 TO 13
2030      K=(I-1)*13+J
2040      SEME(K)=I
2045     VALOR(K)=J
2060    NEXT J
2070 NEXT I
2075  SEME(53)=0 : VALOR(53)=0
2080  RETURN
3000 ' elimina una carta dal mazzo
3010  FOR N=X TO 52
3030     VALOR(N)=VALOR(N+1)
3035     SEME(N)=SEME(N+1)
3040  NEXT N
3070  RETURN
```

```
 GIOCATORE Numero  1
F  :9 1
C  :4 5 8
Q  :7 K 1 9
F  :4 8 10 K
 GIOCATORE Numero  2
P  :10 7 6 J K
C  :1 10 7 K
Q  :J
F  :5 3 1
 GIOCATORE Numero  3
P  :2 5 8 Q
C  :3 J
Q  :8 10 6 4 Q
F  :Q 2
 GIOCATORE Numero  4
P  :4 3
C  :Q 2 6 9
Q  :3 2 5
F  :7 9 J 6
Break in 250
Ok
```

Ecco l'esempio di una smazzata del programma SMAZZ. Chi scriverà il proprio programma in BASIC copiando il testo qui presentato in fig. 1, ben difficilmente riuscirà ad ottenere esattamente questa smazzata: c'è la stessa probabilità che si ottenga-no 2 distribuzioni successive o identiche di carte!

IL LINGUAGGIO MACCHINA

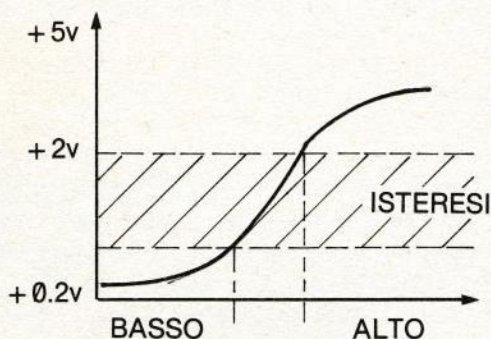
Per imparare a conoscere il linguaggio macchina è utile puntualizzare i concetti di elettronica digitale, TTL, sistema binario e sistema esadecimale. È opportuno, inoltre, approfondire la conoscenza del microprocessore. A questa materia dedichiamo la prima lezione.

1.1 L'ELETTRONICA DIGITALE

Prima di avventurarci nel cuore vero e proprio del microprocessore (il cervello del computer), penso sia utile, per chi affronta per la prima volta il l/m, dare qualche fondamentale nozione di elettronica digitale, che servirà in seguito a rendere più semplice e completo l'apprendimento del linguaggio macchina vero e proprio. Esistono principalmente due campi dell'elettronica, quello digitale e quello analogico. Mentre nell'elettronica analogica si considerano le variazioni di differenze di potenziale (i famosi volts), in elettronica digitale si hanno due sole ben definite differenze di potenziale; vale a dire, in elettronica analogica un dispositivo può generare un campo continuo di volts (per es. da Ø a 2Ø volts), mentre un dispositivo digitale può generare solo due ben definite differenze di potenziale, che variano a seconda della famiglia cui il dispositivo appartiene, ma che all'interno di tale famiglia sono ben definite e standard. Per fare un esempio possiamo dire che un integrato digitale (quei strani contenitori plastici neri con due file di piedini sporgenti che si trovano all'interno di ogni computer e che contengono uno o più dispositivi digitali) della famiglia dei TTL, può generare solo due valori di tensione in uscita, Ø volts oppure +5 volts. Naturalmente esistono più famiglie di integrati digitali quali DTL, ECL, TTL, RTL ecc. L'appartenenza di un integrato ad una data famiglia dipende esclusivamente dall'architettura interna e dagli standard di tensione necessari al dispositivo stesso, ma evitiamo questo discorso, che sarebbe troppo vasto e complesso da spiegare in poco spazio.

1.2 LA FAMIGLIA DEI TTL

La famiglia di integrati che possono principalmente interessarci è quella dei TTL (Transistor-Transistor Logic), in quanto questa è la più usata nella progettazione ed interfacciamento di computers. Abbiamo già accennato al fatto che un dispositivo TTL può generare un segnale di Ø volts o +5 volts, ma questo avviene solo in teoria, in quanto a causa di vari fattori interni ed esterni, tali

tensioni possono variare di una piccola percentuale. Quello che comunque ci interessa maggiormente è il fatto che il dispositivo è in grado di generare due tensioni ben separate tra loro da un intervallo che viene chiamato 'isteresi' del dispositivo.
Nel grafico si nota come l'impulso di tensione che in teoria dovrebbe essere di +5 volts (tale livello lo chiamiamo 'alto') possa variare da un minimo di +2 volts ad un massimo di +5 volts, mentre l'impulso di tensione che dovrebbe essere di 0 volts (questo livello lo chiamiamo 'basso') può arrivare ad un massimo di +0.8 volts.
Se noi al livello basso di tensione attribuiamo il numero 0 ed al livello alto il numero 1, otteniamo che un dispositivo digitale è in grado di generare un numero che ha valore 0 o 1, che viene chiamato 'BIT'. Questa è appunto la spiegazione per cui tutti i computers si basano sul sistema di numerazione binario, in quanto questo permette operazioni basandosi su numeri formati da due soli simboli, 0 oppure 1.

1.3 IL SISTEMA BINARIO
Ogni sistema di numerazione è caratterizzato da un numero b, detto base del sistema, che equivale al numero di simboli disponibili nel sistema dato per rappresentare un numero. Nel sistema decimale, b equivale a 10, in quanto abbiamo a disposizione dieci simboli (da 0 a 9), le cui combinazioni ci permettono di rappresentare qualsiasi numero. Una regola da tener ben presente è che se si hanno n cifre formanti un numero, le combinazioni possibili tra queste n cifre sono uguali alla base b elevata ad n (b^n), infatti in base decimale con quattro cifre noi possiamo rappresentare $10^4 = 10000$ combinazioni diverse di numeri (da 0000 a 9999). Inoltre una di queste combinazioni, per esempio 3658, è scomponibile nel seguente modo:

$3658 = 3 \times 1000 + 6 \times 100 + 5 \times 10 + 8$

ossia:

$3658 = 3 \times 10^3 + 6 \times 10^2 + 5 \times 10^1 + 8 \times 10^0$

Riconsiderando la b e la n della discussione precedente notiamo che $10^2 = 10^{h-1}$, $10^2 = 10^{h-2}$, e così via; per cui:

$3658 = 3 \times 10^{h-1} + 6 \times 10^{h-2} + 5 \times 10^{h-3} + 8 \times 10^{h-4}$

Consideriamo ora le cifre 3,6,8,5, come coefficienti, e chiamiamoli a. Ne segue che un qualsiasi numero A è rappresentabile con la seguente formula:

$A = a_{h-1} \times b^{h-1} + a_{h-2} \times b^{h-2} + ...a_1 \times b^1 \times a_0 \times b^0$

Quest'ultima formula (polinomiale) è applicabile a qualsiasi sistema di numerazione, per cui un numero binario di 4 cifre (1011) è scomponibile come:

$1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

Tale calcolo risolto dà come risultato 11 in base decimale (11_{10}). Possiamo così convertire facilmente una cifra da binario a decimale, e di conseguenza un dispositivo digitale può rappresentare facilmente anche un numero decimale. Se per esempio abbiamo tre diverse linee elettriche, ed in ognuna di queste abbiamo o 0 volts o +5 volts abbiamo una combinazione binaria. Infatti se 0 volts = 0 e +5 volts = 1 segue che:

+5V ______ 1
+0V ______ 0
+0V ______ 1

Da ciò segue che tre linee elettriche possono rappresentare, come da figura, il numero binario 101_2 (101_2 per dire 101 in base 2), equivalente al numero decimale 5_{10} (5_{10} per dire 5 in base 10). Anche in questo caso, come nel sistema decimale, date n cifre si possono calcolare il numero di combinazioni possibili elevando la base b ad n,b. Per cui in un sistema binario con tre cifre si possono avere un massimo di $2^3 = 8$ combinazioni, con quattro cifre un massimo di $2^4 = 16$ combinazioni.

1.4 IL SISTEMA ESADECIMALE

Imparato a portare un numero a binario a decimale, sorge ora un problemino; infatti un programmatore che per la prima volta si avventura nel linguaggio macchina, nota che un numero decima-

le è rappresentabile tramite quattro bits, ma le combinazioni di tali unità binarie sono $2^4 = 16$, quindi ben al di sopra delle dieci che richiede il sistema decimale, infatti rappresentando tutte le combinazioni di quattro bits e mettendovi a fianco la corrispondente conversione decimale otteniamo:

binario	decimale
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	?
1011	?
1100	?
1101	?
1110	?
1111	?

(figura 1.4/1)

Si vede quindi che non esiste alcun simbolo per rappresentare le combinazioni superiori a nove (qualcuno certo penserà a 10,11,12 ..., ma ricordo che questi sono numeri e non simboli).
Da ciò si rende necessaria l'introduzione di altri simboli, che insieme ai dieci simboli decimali costituiranno il sistema esadecimale (a base 16); tali simboli sono le prime sei lettere dell'alfabeto (A-F), per cui si ottiene la seguente tabella:

binario (base 2)	decimale (base 10)	esadecimale (base 16)
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	'10'	A
1011	'11'	B
1100	'12'	C

1101	'13'	D
1110	'14'	E
1111	'15'	F

(figura 1.4/2)

Se qualcuno ancora pensasse che i numeri 10,11,... ecc. avessero risolto tale problema altrettanto bene quanto l'esadecimale, faccio notare che se abbiamo un numero binario a otto bits, noi lo possiamo suddividere in due numeri in due numeri da 4 bits ciascuno (nibble).
per esempio:
10110010 si può dividere in 1011 e 0010.

Rappresentando questi due nibbles con due numeri decimali risulterebbe:

$1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11_{10}$
$0010 = 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 2_{10}$

Accoppiando in seguito i due numeri ottenuti si ha come risultato 112.
Se calcoliamo ora il vero valore del numero ad 8 bits otteniamo.

$10110010 = 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^0 = 178_{10}$

Notiamo quindi che in decimale il numero ricavato dai nibbles (112) non corrisponde al reale valore del byte. Se usiamo il sistema esadecimale sui nibbles otteniamo il numero B2H(H stà per esadecimale).
Ricavandoci ora con la formula polinomiale il corrispondente valore decimale di B2 otteniamo:

$B2_{16} = B_{10} \times 16^1 + 2_{16} \times 16^0$

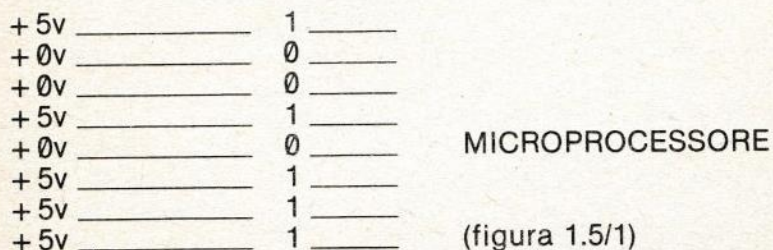
Ricordando che B_{16} corrisponde al valore 11_{10} (fig. 1.4/2) segue:

$B2_{16} = 11 \times 16^1 + 2 \times 16^0 = 178_{10}$

che corrisponde al valore effettivo del byte.
Quindi col sistema esadecimale noi possiamo rappresentare direttamente numeri binari formati da multipli di 4 bits tramite simboli, mentre col sistema decimale noi dovremmo ricavarci il valore del numero tramite formula aritmetiche.

1.5 I MICROPROCESSORI

Il menzionare bytes, bits e TTL non è stato fino ad adesso senza scopo, infatti tutto ciò serve ad introdurvi al uP (Microprocessore) ad un livello superiore a quello raggiunto da coloro che hanno esperienza nel solo campo informatico. Il mio scopo è quello di farvi capire oltre il linguaggio macchina le sue interconnessioni con la parte 'hard' (la parte tecnica vera e propria).
Un uP ha il compito di svolgere tutte le funzioni matematiche e di trasmissione dati all'interno di un computer. Esternamente il uP si presenta come un normale integrato TTL di notevoli dimensioni, il cui contenitori (DIP), oltre che plastico, è spesso anche ceramico. All'interno di tale contenitore esiste il 'chip', una piastrina di silicio di piccolissime dimensioni (qualche mm quadrato). Tale chip è connesso elettricamente ai piedini (pins) che escono ai due lati del DIP. Tali piedini hanno la funzione di collegare elettricamente l'integrato stesso al circuito stampato (la piastra con piste in rame su cui sono montati i componenti) in modo da creare varie connessioni tra tutti gli integrati. L'insieme di tali connessioni è detto 'rete logica' e permette la trasmissione dei dati e la sincronizzazione tra i vari integrati o dispositivi esterni (periferiche).
Ogni pin di un uP svolge varie funzioni. Una parte dei piedini di un uP è dedicata a ricevere ed a trasmettere dati o istruzioni da e per la CPU (un altro modo di definire il microprocessore, da Central Processing Unit). Tale insieme di piedini è chiamato 'Data Bus' (via per i dati) ed il funzionamento è abbastanza semplice. Poniamo l'esempio che tale 'Bus' sia formato da 8 pins (in tal caso il uP si dice ad 8 bits) e forniamo a tali piedini una combinazione di livelli alti e bassi (0 volts e +5 volts). Ogni combinazione che noi forniamo sul bus può corrispondere ad un dato numerico o istruzione che la CPU deve elaborare. Spiegando ciò tramite uno schema otteniamo la figura 1.5/1, in cui troviamo 8 linee elettriche (il Data Bus) che vanno a connettersi al uP.

```
+5v ____________ 1 ______
+0v ____________ 0 ______
+0v ____________ 0 ______
+5v ____________ 1 ______
+0v ____________ 0 ______     MICROPROCESSORE
+5v ____________ 1 ______
+5v ____________ 1 ______
+5v ____________ 1 ______     (figura 1.5/1)
```

La combinazione dei segnali dà come risultato un numero binario, (10010111 in fig. 1.5/1) che per il uP ha un significato ben preciso e che può essere un dato numerico su cui lavorare o un'i-

struzione da eseguire. Da ciò si può ricavare che:

1) Il uP (di conseguenza il linguaggio macchina) non fà distinzioni tra dati numerici su cui lavorare o istruzioni da eseguire, infatti entrambi sono combinazioni binarie.

2) Il linguaggio macchina si riduce sempre ad un insieme di numeri, spetta al programmatore ordinarli in modo che il uP esegua il programma in modo corretto (vedremo poi come).

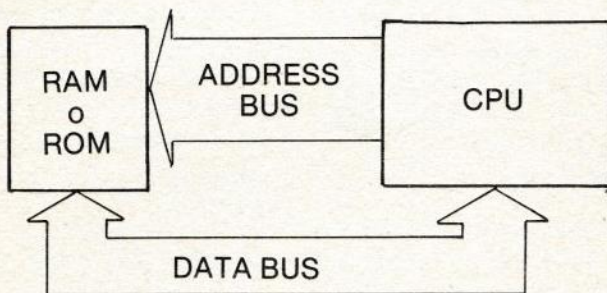
Per rendere più evidente di come i numeri possano rappresentare delle istruzioni prendiamo ad esempio il numero 76_{16} (118_{10}). Tale numero, inviato sul bus dati di una CPU Z-80 (Z-80 è il nome), ferma ogni elaborazione di quest'ultima, la quale rimane in attesa che avvenga una certa condizione, oppure il dato $F8_{16}$ (243_{10}), sempre su Z-80, impedisce che la CPU venga interrotta da un dispositivo esterno durante una elaborazione dati.
Naturalmente la corrispondenza tra codici numerici ed istruzione non è universale per tutti i uP, ma varia a seconda del tipo della CPU. Inoltre i dati inviabili su una data bus di un uP varia a seconda delle dimensioni del bus stesso, da qui la denominazione di un uP a 8,16,32 bits, a seconda, appunto, l'accettare le combinazioni di 8,16,32 linee elettriche in entrata come dati.
Ecco i nomi di alcuni tra i più usati microprocessori:

6510	8 bits	
Z-80	8 bits	
8080	8 bits	
8085	8 bits	(figura 1.5/2)
68000	16 bits	
TMS 9900	16 bits	
Z 8000	16 bits	

A questo punto, capito come si invia una istruzione ad un uP bisogna spiegare come quest'ultimo può trovare la sequenza di istruzioni da eseguire. Oltre al 'Data Bus', esiste un 'Address Bus' la cui costituzione è identical al 'Data Bus', ma dove la direzione dei dati è solo verso le memorie. Le combinazioni di Bits su questo bus, invece di rappresentare delle istruzioni, vanno a pilotare delle memorie esterne. Queste sono formate da un insieme di piccole celle che hanno la capacità di memorizzare un byte e che sono numerate. Le varie combinazioni sull'Address bus servono a selezionare una di queste celle. Una volta selezionata una cella, il contenuto di questa è posto dalla memoria sul Bus dati, permettendo alla CPU di leggerlo. Naturalmente la CPU, una volta selezionata una cella, può anche scrivervi all'interno, ed è compito della memoria raccogliere il dato da memorizzare dal Data bus. Notiamo quindi una differenza tra Data ed Address

bus. Infatti nel primo i dati possono venire trasmessi dalla memoria alla CPU e vice versa. Per tal motivo tale bus è detto 'Bidirezionale' (con due direzioni di flusso dati), al contrario dell'Address bus, che si dice 'unidirezionale', in quanto il flusso di indirizzi va solo verso la memoria. Ricordiamo che l'Address bus può variare in grandezza (come il Data bus) a seconda del uP cui appartiene. Usualmente un uP ad 8 Bits ha un Address bus di 16 Bits, la cui combinazione può indirizzare $2^{16} = 65536$ celle di memoria diverse. Tale numero è anche noto come 64K, facendo corrispondere 1K a $2^{10} = 1024$ celle di memoria. Dunque più è ampio un Address bus, maggiore è la capacità di memoria accessibile da una CPU.
Ecco a questo punto le dimensioni degli Address Buses dei precedenti microprocessori:

uP	Data Bus	Addr.Bus
6510	8 bits	16 bits
Z-80	8 bits	16 bits
8080	8 bits	16 bits
8085	8 bits	16 bits
68000	16 bits	24 bits
TMS 9900	16 bits	16 bits
Z-8000	16 bits	16/23 bits

Inoltre eccovi uno schema flusso dati delle connessioni tra memoria esterna e CPU:

La freccia dell'Address bus indica un unico flusso dati (gli indirizzi) verso la memoria, mentre quella doppia della Data bus indica le due possibili direzioni del flusso di dati.
Detto questo concludo questa prima parte sul l/m, sperando di essere stato abbastanza chiaro con questa introduzione elettronica, rendendovi forse per adesso l'apprendimento un po' complesso, ma sicuro di esservi stato utile per futuri capitoli del corso sul l/m e su tutti i uP in generale. 1 - continua

A.V.S.

LE FUNZIONI PRINCIPALI

La volta scorsa abbiamo esaminato le istruzioni fondamentali del BASIC e abbiamo anche parlato di variabili indicizzate.
Stavolta ci occuperemo invece di alcune istruzioni di uso meno frequente e di tutte le funzioni di cui dispone il BASIC dello Spectrum.
Come certo ricorderete, per porre il computer in condizione di ricevere dati durante l'esecuzione di un programma abbiamo sempre usato l'istruzione INPUT, ma essa non è l'unica che permette di interagire con l'operatore: esiste infatti anche l'istruzione INKEY$, la quale però, diversamente da INPUT, non riceve un insieme di dati seguiti dalla pressione di ENTER, ma "legge" quale tasto è stato premuto al momento dell'esecuzione dell'istruzione ed eventualmente ne assegna il valore ad una variabile. Per esempio, l'istruzione:

```
LET A$ = INKEY$
```

riconosce quale tasto è stato premuto e ne assegna il carattere corrispondente alla variabile A$. Se nessun tasto era premuto al momento dell'esecuzione di questa istruzione, alla variabile A$ sarà assegnato un valore nullo.

Per restare in tema di variabili, argomento peraltro assai ampio, citiamo un altro metodo di assegnazione di valori ad una variabile, stavolta però interno al programma ed in cui i dati interessati devono essere scritti durante la stesura del programma stesso. L'istruzione a cui ci riferiamo è READ che in inglese significa proprio LEGGI.
Se in un programma dobbiamo usare grandi quantitativi di dati che rimangono sempre tali e non devono quindi essere frequentemente aggiornati, possiamo introdurre tali dati in un programma grazie all'istruzione DATA seguita appunto dai dati che ci interessano i quali possono essere numerici o alfanumerici e separati fra loro da una virgola. Questi dati saranno poi letti in sequenza dall'istruzione READ seguita da un nome di variabile a cui si desidera assegnare il dato letto; fate attenzione che il tipo di variabile corrisponda effettivamente al tipo di dato che deve leggere; inoltre ricordate che le stringhe devono essere chiuse fra apici.

Vediamo ora un semplice esempio:

```
10 FOR A = 1 TO 7
20 READ S$
30 PRINT S$
40 NEXT A
50 DATA "LUNEDI",
   "MARTEDI"
60 DATA "MERCOLEDI",
   "GIOVEDI"
70 DATA "VENERDI",
   "SABATO"
80 DATA "DOMENICA"
```

Come probabilmente avrete ca-

pito, la linea 10 inizia un ciclo che va da 1 a 7, all'interno del quale si trova una istruzione che legge (READ) i valori contenuti nelle linee di DATA e li assegna alla variabile stringa S$ che viene stampata subito dopo. Come potete vedere, nelle linee di DATA che seguono il programma principale abbiamo inserito il nome dei giorni della settimana che sono un tipico esempio di dato costante; per chiarezza abbiamo inserito solo due giorni per ogni istruzione DATA, ma è possibile porne molti di più. Sconsigliamo comunque di farlo perché più lunga è una linea e più tempo il computer impiega a spostare il cursore all'interno di essa. Non ha comunque nessuna importanza se i dati si trovano su linee diverse, in quanto l'istruzione READ punta sempre al dato successivo a quello appena letto, ovunque esso si trovi.
Se dovessimo però leggere più volte gli stessi dati non potremmo farlo se non ricorrendo ad una speciale istruzione chiamata RESTORE, la quale "resetta" i puntatori al primo dato della prima istruzione DATA, permettendo quindi di rileggere il tutto quante volte vogliamo. Vediamo un semplice esempio:

```
10 PRINT "NUMERI PRIMI"
20 FOR A=1 TO 5
30 READ N
40 PRINT N
50 NEXT A
60 LET P$=INKEY$
65 IF P$="" THEN GOTO 60
70 RESTORE:CLS
75 GOTO 10
90 DATA 1,3,5,7,11
```

Questo programmino stampa i cinque numeri primi che sono memorizzati in linea 90, con un ciclo simile al precedente; le linee 60 e 65 attendono la pressione di un tasto per fare in modo che vengano eseguite le linee 70 e 75 che "resettano" i puntatori di dato e cancellano lo schermo, tornando quindi alla linea 10.

Concludiamo osservando che le linee di DATA non si devono trovare necessariamente alla fine ma, poiché non eseguono nessuna operazione, sarebbe inutile porle nel mezzo del programma poiché causerebbero solo un rallentamento nei tempi di esecuzione; è inoltre preferibile che siano inserite alla fine per ovvie ragioni estetiche e per non pregiudicare la facile lettura del programma da parte di un'altra persona.
Ora che avete imparato a creare dei programmi BASIC abbastanza impegnativi, dovreste imparare anche a "salvarli" e "caricarli" su una normale musicassetta.

Se avete in memoria un programma, non dovete fare altro che introdurre nel registratore collegato al vostro computer una cassetta vergine (o comunque che non usate) e scrivere poi il comando SAVE "nome", dove con "nome" si intende una stringa composta al massimo da 10 caratteri che rappresentano il nome del programma. Battete ora il tasto ENTER e vedrete apparire la scritta:

START TAPE THEN PRESS A KEY

ovvero:

AVVIA IL REGISTRATORE E PREMI UN TASTO

Eseguite premendo contemporaneamente RECORD e PLAY sul registratore e poi schiacciando un tasto; vedrete delle righe scorrere sui bordi dello schermo: ciò significa che il computer sta salvando su nastro il programma BASIC presente in memoria.
Terminata questa operazione apparirà il messaggio Ø:OK. A questo punto sarebbe sempre meglio verificare la corretta registrazione dei dati (per evitare spiacevoli sorprese) usando il comando VERIFY "nome", il quale confronta ciò che è presente su nastro con i dati in memoria e segnala eventuali errori (in questo caso ripetere l'operazione di salvataggio); se tutto è a posto apparirà il messaggio OK. Ora il vostro programma è memorizzato permanentemente sulla cassetta e potrete ricaricarlo in memoria quando vorrete con il comando LOAD "nome". All'esecuzione di questo comando il bordo dello schermo cambierà colore e, avviando il registratore, lampeggerà finché non troverà un segnale (rappresentato dalle strisce colorate); allora stamperà sul video il nome del programma trovato e se esso coincide con quello da voi specificato nel comando LOAD, procederà al caricamento. Fate molta attenzione durante il salvataggio dei dati, perché il cavetto EAR deve essere scollegato da almeno una delle due parti; in caso contrario il programma non verrà salvato.
Se volete ottenere che un programma parta da solo appena caricato, dovrete specificare il numero di linea alla quale dovrà iniziare l'esecuzione, direttamente nel comando SAVE. La sintassi per definire la linea di autostart in un comando SAVE è la seguente:

SAVE "nome" LINE N

dove n è il numero di linea alla quale inizierà l'esecuzione del programma.
Con lo Spectrum è anche possibile salvare matrici di variabili numeriche o alfanumeriche o parti di memoria sotto forma di bytes. Se in un programma BASIC decidiamo di usare una variabile indicizzata per archiviare qualcosa (per esempio gli indirizzi degli amici) potremo poi salvare su nastro tutta la matrice e poi ricaricarla, modificarla e risalvarla a nostro piacimento. L'istruzione per salvare una matrice è:

SAVE "nome" DATA x()

dove x è il nome della matrice (seguita dal $ se si tratta di stringhe). Le procedure di verifica e caricamento sono le seguenti:

VERIFY "nome" DATA x()
LOAD "nome" DATA x()

Ricordatevi comunque di dimensionare la matrice interessata prima di caricarla da nastro.
Terminiamo con qualche accenno alle funzioni di editing (correzione) disponibili sullo

Spectrum. Se avete un programma BASIC in memoria potrete vederlo con il comando LIST, il quale fornisce il LISTATO (così si chiama l'insieme delle istruzioni e numeri di riga). Ogni volta che lo schermo viene riempito, il computer vi porrà la domanda SCROLL?; se non desiderate che il computer prosegua l'esame del listato premete il tasto SPACE o N, altrimenti un qualsiasi altro tasto. Per richiamare una riga desiderata scrivete:

LIST N

dove N è la linea interessata; quindi premete SPACE se il computer vi chiede lo scrolling e poi premete contemporaneamente i tasti CAPS SHIFT e 1: in questo modo la riga che vi interessa sarà riportata nella parte bassa del video e potrete apportarvi le modifiche che desiderate. Per spostarvi a destra o a sinistra usate i tasti 5 e 3 contemporaneamente a CAPS SHIFT, mentre per cancellare il carattere alla sinistra del cursore usate CAPS SHIFT e 0. Premete ENTER per reinserire la linea al suo posto. Per maggiori informazioni sulle funzioni di editing è comunque consigliabile consultare il manuale della Sinclair.

Passiamo ora a descrivere dettagliatamente tutte le funzioni che il BASIC dello Spectrum mette a disposizione.

La prima che incontriamo è ABS, la quale restituisce il valore assoluto del numero o variabile numerica usata come argomento; in altre parole priva il numero dell'eventuale segno che lo precede. Per esempio l'istruzione:

PRINT ABS —91.4

produrrebbe la stampa di 91.4. Questa particolare funzione può rivelarsi utile nel caso in cui si desideri sapere se una variabile contiene un numero positivo o negativo; basta infatti procedere nel modo seguente:

IF A = ABS A THEN...

In questo modo le istruzioni che seguono il THEN saranno eseguite solo se A contiene un valore positivo. Cercate di capirne da soli la ragione.

La funzione SGN determina invece direttamente il segno di una variabile posta fra parentesi e fornisce come risultati:

1 se positivo
—1 se negativo
0 se nullo

Sapendo questo, possiamo riscrivere la condizione dell'esempio precedente nella seguente forma:

IF SGN A = 1 THEN...

Infatti, il risultato della funzione sarà 1 solo se la variabile contiene un numero positivo.

Un'altra funzione matematica molto usata è INT che fornisce la parte intera del numero dato (l'arrotondamento è sempre effettuato all'intero più basso); quindi l'istruzione:

PRINT INT 10.9

darebbe come risultato 10 mentre l'istruzione:

PRINT INT —20.1

restituirebbe il valore —21. Questa particolare funzione è spesso usata in abbinamento a RND che permette di generare numeri casuali o pseudocasuali. Il numero generato da questa funzione è sempre compreso fra 0 e 1, ma non sarà mai 1. Con qualche artificio di programmazione è comunque possibile produrre sequenze di numeri compresi entro i limiti che ci interessano; infatti, se moltiplichiamo il risultato della funzione RND per il massimo numero che ci interessa, otterremo proprio numeri compresi fra 0 e il numero specificato. Vediamo un esempio:

```
A= RND *20
```

In questo modo otterremo che la variabile A contenga solo numeri compresi fra 0 e 19. Ma se desideriamo che la sequenza di numeri non inizi da 0 ma da un altro valore a nostro piacere, basta sommare il numero più basso della serie al risultato della funzione RND moltiplicato per l'intervallo esistente fra il minimo e il massimo. Ci affrettiamo a chiarire tutto con un semplice esempio; se volessimo generare solo numeri casuali compresi fra 50 e 80 procediamo nel modo seguente:

```
A= 50+ RND*(80—50+1)
```

Questa formula è sempre valida e si potrà rivelare molto utile in numerose occasioni. Notate che alla differenza dei due limiti deve essere aggiunto 1 altrimenti la funzione arriverebbe al massimo a 79. Fate però attenzione, perché in questo modo saranno generati numeri reali: quindi, se siete interessati solo agli interi sarà bene che facciate precedere il tutto dalla funzione INT che abbiamo appena visto, riscrivendo la linea come segue:

```
A= INT (50+RND*31)
```

Dove 31 non è altro che la differenza fra 80 e 50 incrementata di 1 unità.
Lo Spectrum dispone inoltre di una funzione molto comoda che consente di convertire numeri binari in decimali; la funzione è BIN. Se noi scrivessimo:

```
PRINT BIN 110
```

otterremmo la stampa di 6.
Oltre a tutte queste funzioni di uso generale, questo computer dispone di una serie completa di funzioni trigonometriche e matematiche; queste sono: SIN per il seno, COS per il coseno, TAN per la tangente, ATN per l'arcotangente, ASN per l'arcoseno, ACS per l'arcocoseno, LOG per i logaritmi, EXP per l'elevamento a potenza e SQR per le radici quadrate.
La maggior parte di queste funzioni non richiedono particolari commenti, poiché si limitano a ricavare il valore di una determinata funzione matematica a partire dal valore che noi forniamo. L'unica che merita qualche precisazione è la funzione EXP, la quale restituisce il valore dell'elevamento a potenza della costante 2.71828183; l'istruzione:

```
PRINT EXP 7
```

equivale quindi a scrivere:

```
PRINT 2.71828183 ↑ 7
```

Tutte queste funzioni non sono comunque di uso molto frequente e potrebbero interessare solo a chi intendesse usare il proprio computer per applicazioni matematiche.
Passiamo ora alle funzioni per il trattamento delle stringhe che lo Spectrum può manipolare con estrema facilità.
La funzione CODE fornisce come risultato il codice del primo carattere della stringa o variabile alfanumerica posta fra parentesi (per conoscere i codici corrispondenti ai caratteri si consulti una tabella); quella opposta a CODE è CHR$ che fornisce il carattere corrispondente al codice numerico usato come argomento.
Una funzione molto utile è LEN, che permette di conoscere la lunghezza della stringa in esame. Per esempio, se in un programma definiamo una variabile alfanumerica, possiamo conoscere di quanti caratteri è composta applicando questa funzione. Vediamo un esempio:

```
10 LET A$ = "COMPUTER"
20 PRINT LEN A$
```

In questo modo otterremo la stampa di 9 che è appunto la lunghezza della stringa A$.
Veniamo ora al punto forte dell'elaborazione delle stringhe: l'estrazione e la manipolazione di parti costituenti una stringa principale. La funzione che ci permette di trattare in modo completo le stringhe è TO (basta infatti specificare il primo e l'ultimo carattere da estrarre). Riferendoci alla stringa A$ precedentemente definita, se scrivessimo:

```
PRINT A$ (1 TO 4)
```

otterremmo come risultato la stringa COMP che corrisponde appunto ai caratteri da 1 a 4 della stringa principale COMPUTER. In questo caso, poiché il primo carattere da estrarre è il primo della stringa, possiamo omettere il punto di partenza scrivendo solo A$ (TO 4).
Se invece l'ultimo carattere da estrarre fosse anche l'ultimo della stringa principale, potremmo ometterne il numero scrivendo:

```
PRINT A$ (5 TO)
```

con cui produrremmo la stampa di UTER. Lo stesso risultato si avrebbe scrivendo PRINT A$ (5 TO 8).
Se si desidera estrarre un solo carattere è sufficiente specificarne la posizione all'interno della stringa principale; se scriviamo:

```
PRINT A$ (3)
```

vedremo apparire la lettera M che rappresenta proprio il terzo carattere all'interno della stringa A$.
Bene, anche questa puntata è terminata. Nel prossimo numero ci occuperemo delle istruzioni per la gestione della grafica e di altre funzioni particolari.

Massimo Cellini

"COME TRADURRE UN NUMERO DECIMALE ALLA BASE 2 O 16"

I numeri in decimale ci sono familiari. Si dividono in 2 categorie: i numeri interi (senza virgola) e quelli con la virgola.
Ora vediamo come convertire un numero decimale (con o senza virgola!) in un numero di qualunque base compresa tra 2 e 16.
Il solito programma BASIC vi aiuterà ad approfondire i concetti esposti.

RAPPRESENTAZIONE DEI NUMERI

Nello scorso numero abbiamo parlato dei "Sistemi di Numerazione" decimale, binaria, ottale ed esadecimale.
Dopo esservi esercitati con il programma basic nelle diverse rappresentazioni, ora siete sicuramente in grado di destreggiarvi da un sistema di numerazione all'altro e magari avete anche imparato ad usare in modo più consapevole anche la familiare "rappresentazione decimale".
L'**informazione** all'interno dell'elaboratore è rappresentata in binario, all'esterno è invece rappresentata in decimale: il passaggio dell'informazione tra l'interno e l'esterno di un elaboratore è garantito dalle unità di ingresso ed uscita, che provvedono a convertire l'informazione da rappresentazione binaria a decimale e viceversa (di questi sistemi, noti come CODICI, ne parleremo in seguito).

Quando diventerete esperti programmatori vi capiterà d'aver bisogno di leggere alcune informazioni direttamente nella rappresentazione interna, per rimuovere errori od anche solo per curiosità.
Vediamo allora come è possibile passare dalla rappresentazione decimale alla rappresentazione binaria (il viceversa siete già in grado di farlo*).
Per convertire un numero intero decimale nel corrispondente numero binario (ottale od esadecimale) si utilizza il seguente metodo.

Metodo della divisione. Si divide il numero intero decimale per la "base" del sistema di numerazione a cui vogliamo passare (2, 8 o 16); si pone il resto della divisione come cifra binaria (ottale od esadecimale) meno significativa (da destra verso sinistra) del corrispondente numero binario che stiamo costruendo. Si divide poi il risultato della prima divisione ancora

per la "base" e si ripete il processo, ottenendo via via le altre cifre.
Il processo si arresta quando il risultato di una divisione diventa uguale a 0.
Vogliamo, per esempio, rappresentare il numero 183_{10} in binario, ottale ed esadecimale: di seguito riportiamo i "**risultati parziali**" delle divisioni ed il **resto**

BASE 2

VALORE INIZIALE E RISULTATI PARZIALI	RESTO
183 : 2	
91	1
45	1
22	1
11	0
5	1
2	1
1	0
0	1

$183_{10} = (10110111)_2$

BASE 8

	RESTO
183 : 8	
22	7
2	6
0	2

$183_{10} = (267)_8$

BASE 16

183 : 16	
11	7
0	B

$183_{10} = (B7)_{16}$

Abbiamo fin qua visto come trasformare un numero intero decimale in un numero binario (ottale ed esadecimale); ma come ben sapete i numeri interi non esauriscono l'insieme dei numeri che abitualmente usiamo: ci sono anche i numeri frazionari (es. 10.5). Di seguito analizzeremo come trasformare questi numeri; per quanto riguarda la parte intera del numero (es. 10.5, parte intera = 10, parte decimale = 5) viene trasformata secondo il metodo della divisione, illustrato nelle pagine precedenti; mentre la parte decimale viene convertita con il "**metodo della moltiplicazione**".

Questo metodo si può schematizzare come segue.
Si moltiplica la parte frazionaria del numero per 2 (8 o 16); si prende la parte intera del prodotto ottenuto e lo si pone come cifra binaria (ottale od esadecimale) più significativa (**attenzione** in questo metodo da sinistra verso destra) del corrispondente numero binario; si procede in modo iterativo moltiplicando la parte decimale del prodotto ottenuto, arrestandosi quando si raggiunge la precisazione desiderata.

Osservate che questo metodo non si arresta sempre a zero, come il precedente, ma siamo noi a dover decidere di arrestare la rappresentazione del numero alla terza, quarta...cifra decimale, cioè all'ultima **cifra** che riteniamo **significativa**.
Consideriamo il numero 0.250 di cui vogliamo la rappresentazione binaria; questo numero ha 3 cifre nella parte fraziona-

ria, decidiamo allora di arrestare il metodo della moltiplicazione anche alla terza cifra binaria (o prima se riscontriamo lo zero)

0.250	.	2	=	0.5	0
0.5				1.0	1
0.0				0	0

$(0.250)_{10} = (0.010)_2$

Anche in questo numero il programma basic vi aiuterà a capire meglio quanto spiegato nelle pagine precedenti, permettendovi di verificare in ogni momento se il numero binario che avete ottenuto è realmente la conversione del numero decimale di partenza; esercitatevi e sarete allora pronti per affrontare il prossimo argomento. Aritmetica binaria.

IL PROGRAMMA BASIC

Vediamo la realizzazione in BASIC della conversione da DECIMALE ad altra base per numeri interi.
Come limite non convertiamo numeri che nella nuova base abbiamo più di 8 cifre.
Il programma è la realizzazione BASIC del metodo illustrato nelle pagine precedenti. Tale metodo si basa, come abbiamo visto nelle pagine precedenti, su successive divisioni e determinazioni di altrettanti resti. La sequenza dei resti messi uno accanto all'altro (nell'ordine inverso a cui sono generati), è la rappresentazione del numero nella nuova base. Perciò ci si serve di una variabile R(I) per contenere le cifre del numero nella nuova base.
La prima volta il numero da dividere (dividendo) è DEC, cioè il numero introdotto (istruzione 30).
Le volte successive il quoziente della divisione precedente diventa il nuovo dividendo. Ricordiamo che il quoziente è l'intero più grande contenuto nel risultato della divisione, e che il divisore è la base prescelta (istruzione 20).
In sostanza il quoziente si trova con l'istruzione 60 e viene memorizzato in A. (Osservate che la prima volta Q è uguale a DEC). Il resto della divisione è poi calcolato dall'istruzione 70.
L'istruzione 75 imposta il nuovo dividendo (Q) al valore del quoziente della divisione (A).
La sequenza di istruzioni 60-75 vengono ripetute (in virtù di 60, 80 e 90) finché Q non diventa O.
Il risultato viene presentato emettendo le cifre trovate scorrendo R(I) all'indietro.
Ultimo commento: poiché R(I) contiene numeri e non cifre di sistemi in base superiore a 10, è necessario per queste cifre, emettere invece i caratteri A,B, C etcetera come già spiegato nell'articolo precedente.
Questo ci induce a considerare ogni cifra come rappresentata da un carattere e costruire il vettore di conversione S$(I). Osservate che poiché R(I) può valore O, e O non può essere indice (istruzione 110), bisogna ag-

giungere 1 a R(I) per ottenere la cifra corrispondente nel vettore di conversione.

MGD

```
10 DIM R(8),S$(16)
13 S$(1)="0":S$(2)="1":S$(3)="2":S$(4)="3":S$(5)="4":S$(6)="5"
15 S$(7)="6":S$(8)="7":S$(9)="8"
16 S$(10)="9":S$(11)="A":S$(12)="B":S$(13)="C":S$(14)
    ="D":S$(15)="E":S$(16)="F"
20 INPUT " Introduci Base " ; BASE
30 INPUT " Introduci numero intero :" ; DEC
40    Q=DEC:I=1:R(1)=0
50 IF Q=0 GOTO 99
60    A=INT(Q/BASE)
70    R(I)=(Q-(A*BASE))
75    Q=A
80    I=I+1
90   GOTO 50
99   'FINE RIPETI
100  FOR K=I TO 1 STEP -1
110    PRINT S$(R(K)+1);
120 NEXT K
run
 PARTE INTERA DEL NUMERO :? 200
A 12  R  8
A 0  R  12
0C8
Ok
run
 PARTE INTERA DEL NUMERO :? 4096
A 256  R  0
A 16  R  0
A 1  R  0
A 0  R  1
01000
Ok
run
```

* vedi articolo precedente

ISTRUZIONI PER LA CASSETTA COMPUTING VIDEOTECA N. 5

COME INIZIARE

(Lato A; ZX Spectrum)

Per caricare i programmi inserite la cassetta nel registratore e scrivete LOAD "" (le " si ottengono con SYMBOL SHIFT e P), subito dopo premete il tasto ENTER e avviate il registratore in PLAY.

A questo punto basterà seguire le istruzioni che appariranno sul video.

Quando avrete finito di usare un programma spegnete il computer staccando per un momento il cavetto di alimentazione. Battete di nuovo LOAD "" seguito da ENTER per caricare il programma successivo.

SPACE TRAFIC

Ancora una volta ti trovi, perduto con le super veloci astronavi, in uno spazio fantastico e pericoloso popolato di imprendibili fantasmi e acchiappafantasmi impazziti che ti corrono incontro per eliminarti senza alcuna pietà. Per iniziare la "lotta" premi il tasto 1. Potrai agilmente muoverti per schivare i nemici utilizzando il tasto 9 per andare in avanti; il tasto 6 per andare in alto; il tasto 7 per andare in basso. Per fare fuoco col tuo laser premi il tasto Ø. Ricorda, hai solo quattro astronavi a disposizione contro i fantasmi che arrivano da tutte le parti e gli acchiappafantasmi che li seguono più veloci e numerosi! Al termine dello scontro, in cui dovrai cercare di totalizzare, naturalmente, il massimo punteggio, potrai inserire il tuo nome nella lista dei migliori "guerrieri" spaziali (centrando le lettere coi tasti 6 e 7 e battendo il tasto Ø per ottenere la scritta). Ricorda che in questo gioco potrai sfidare i tuoi amici per dimostrare chi sarà il più abile guerriero dello spazio. Al termine della partita potrai controllare il tuo punteggio, o confrontarlo con quello dei tuoi compagni di gioco, coprendo i dodici spazi consentiti per scrivere il nome. Il risultato comparirà automaticamente. E poi via di nuovo nello spazio!

PING PONG

Ti piace il ping pong? Pensi di essere un "campione"? Bene, finalmente potrai dimostrarlo a te stesso e ai tuoi amici in un'avvincente partita giocata sullo schermo del tuo Spectrum.

All'inizio del gioco, dopo la domanda apparsa sullo schermo, potrai scegliere se sfidare il computer (tasto 1) o un altro avversario (tasto 2). Subito dopo ti verrà chiesto di decidere il livello di difficoltà della partita: batti un numero da 1 a 6. Infine inserisci il tuo nome (se sfidi lo Spectrum) o quello dei due giocatori e schiaccia il tasto ENTER.

Sullo schermo apparirà l'immagine delle racchette e della pallina. Per iniziare a giocare usa il tasto B, che dovrai premere ad ogni "battuta" della partita. Il giocatore 1 usa i tasti Q-A per muovere le sue racchette (se giochi contro lo Spectrum, questi saranno i suoi comandi); il giocatore 2 usa i tasti P-ENTER.

La partita si conclude quando uno dei giocatori totalizza 11 punti; allora sullo schermo apparirà il nome del vincitore e il punteggio effettuato.

Alla richiesta: "altra partita?" premi il tasto S per continuare altrimenti, se vuoi cambiare livello di difficoltà, premi N e il gioco ricomincerà dalle prime istruzioni.

Allenandoti da solo potrai in seguito sfidare tutti i tuoi amici, sicuro di essere ogni volta il "campione".

Corri a provare PING PONG e buon divertimento!

CAVERN

Sei l'ultimo soldato in grado di usare la potentissima arma a raggio laser che difende l'unico accesso alla città sotterranea dove si sono rifugiati gli abitanti di NAR 7.

I tuoi nemici sono i TARVAL, degli strani esseri provenienti da un'altra galassia, il cui corpo è per metà quello di una mosca e per il resto quello di una medusa. Ma attento! Sono intelligentissimi e cercheranno a tutti i costi di superare con le loro astronavi la barriera difensiva del tunnel d'accesso alla città.

Distruggili usando il tuo cannone laser molto velocemente (tasto P per fare fuoco) e spostandolo per puntarlo con i tasti Q = su e Z = giù.

Per iniziare a giocare premi il tasto S.

Ricorda che hai a disposizione soltanto 3 "vite", quindi dovrai evitare che i nemici ti vengano addosso per distruggere la tua postazione difensiva. Cerca di totalizzare il massimo punteggio possibile abbattendo le astronavi nemiche e ricorda che quelle verdi valgono 30 punti e quelle bianche 50.

Per ricominciare la battaglia premi un tasto.

3D GRAFF

Questo è un programma scientifico per disegnare una qualsíasi funzione in tre dimensioni $z=(x,y)$. Il risultato sarà generalmente una superficie; differentemente dal vu-3d, in cui veniva rappresentato graficamente un oggetto qualsiasi in tre dimensioni, qui la figura disegnata corrisponde ad una ben determina-

ta funzione matematica. Il programma chiede:
1) definire la funzione
2) definire i limiti
3) angolo di elevazione
4) angolo di rotazione
5) definizione linee
6) risoluzione sugli assi
7) blank linee

Vediamo un esempio:
1) z = f(x,y) = x*x-y*y
2) xmin = -10 xmax = 10
ymin = -10 ymax = 10
A questo punto il programma vi chiederà di attendere.
Se tutto è ok si riparte;
3) questo angolo è riferito all'osservatore e serve per vedere la figura sotto diversi punti di vista.
Elevazione = 23.
4) come sopra.
Rotazione = 18
5) poniamo (per la definizione della griglia):
linee x = 12
linee y = 12
(si consiglia di dare sempre valori simmetrici);
6) più la risoluzione è alta più il programma è lento;
poniamo:
risoluzione x = 13
risoluzione y = 13
7) rispondendo "y" avremo la soppressione delle linee nascoste.
Non consigliamo questa scelta se non è assolutamente necessaria, infatti, disponendo "y", il programma diverrà estremamente lento.
Anche qui, terminato il grafico, premendo C si otterrà una copia sulla stampante.

Esempi di funzione sono:
1) x*x/2 + y*y/3 + 2
2) (x*x-y*y)*(x*x-y*y)
3) x*y + sen(x*y)

E altre ancora.....

SQUARE

È un gioco di abilità che impegna a fondo anche la tua intelligenza.

Sullo schermo del tuo Spectrum appare un quadrato formato da 9 caselle di due diversi colori: verde e viola. Premendo i tasti numerici compresi tra 1 e 9, dovrai far variare i blocchi di colori fino ad ottenere un quadrato tutto viola tranne la casella centrale, che dovrà essere verde. Solo allora avrai vinto e meriterai la "medaglia" che il computer ti offrirà per la tua abilità.

Sullo schermo ti verrà segnalato, ogni volta, il numero dei tentativi che hai già effettuato: cerca di vincere col minor numero di tentativi possibile!

(Lato B: TI 99/4A. Le spiegazioni dei programmi appariranno direttamente in video).

Per caricare i programmi inserite la cassetta nel registratore e scrivete OLD CS1 seguito dal tasto ENTER. Apparirà la scritta REWIND CASSETTE TAPE, THEN PRESS ENTER, premete quindi il tasto ENTER, apparirà ora il messaggio PRESS CASSETTE PLAY THEN PRESS ENTER che significa: premi il tasto PLAY del registratore e quindi ENTER: avviate dunque il registratore in play e premete di nuovo ENTER.

Quando il computer avrà terminato di caricare il primo programma fermate il registratore e scrivete RUN seguito come sempre da ENTER per lanciarne l'esecuzione.

Quando avrete terminato di usare un programma resettate il computer spegnendolo per un momento, dopodiché ripetete le operazioni di caricamento sopradescritte.

CASSAFORTE

Quanto vali come scassinatore di casseforti?

Prova con questo gioco (ma solo con questo gioco!) a sperimentare la tua abilità.

La combinazione della cassaforte è composta da 4 numeri.

Tu, prima che scada il tempo segnato dalla riga gialla e giunga la polizia, hai 9 tentativi per scoprirli, ma solo 9 o ti ritrovi carcerato come se fosse scaduto il tempo.

Usa i numeri sulla tastiera.

Quando hai piazzato un numero giusto nel quadrato giusto, questo rimane bloccato.

Tieni presente che ci possono essere più numeri uguali nella stessa combinazione (es: 2128) e che lo Ø è un numero valido.

Divertiti, ma ricordati che il "Crimine non paga".

MEDIA VOTI

È finito il tempo dei calcoli per sapere le medie dei voti riportati nelle varie materie.

Ora con il computer è più facile: otterrai non solo la media per ogni materia, ma anche la media complessiva di tutte le materie.

Caricato il programma, compare la lista delle materie.

Batti un tasto e poi, materia per materia, inserisci: prima il numero dei voti riportati e batti "ENTER", poi ogni singolo voto, battendo "ENTER" ogni volta.
Terminata l'introduzione dei dati, avrai la "videata" di tutte le medie e poi, battendo un tasto, la media complessiva.

LETTERE ASSASSINE

Solo un gioco? No!
Unisci l'utile al dilettevole; impara a conoscere perfettamente la tastiera e... quando riuscirai a superare il decimo quadro, saprai battere a macchina più velocemente di chiunque altro.
Il gioco, in sé molto semplice, consiste nel salvare una città da un attacco di forze aliene che hanno la forma delle lettere dell'alfabeto.
Neutralizza ogni lettera che compare sul video, prima che colpisca un palazzo, schiacciando il corrispondente tasto sulla tastiera.
Il gioco consta di 10 livelli di abilità differenti; la velocità di attacco delle lettere è, naturalmente, direttamente proporzionale al livello di gioco: 1, lento - 10, velocissimo.
Il superamento di un livello ti porta, automaticamente, al livello successivo.
All'inizio della partita hai 31 case da salvare. Il gioco termina con la distruzione dell'ultima.
Ogni casa salvata ti frutta 10 punti. Passando da un livello di gioco all'altro le case perse in precedenza non ti vengono restituite.
Dato il Run, il computer ti mostra le istruzioni: premi un tasto, scrivi ora da quale livello vuoi partire e batti "ENTER". Alla fine di ogni partita il computer ti chiede se vuoi giocare ancora o no.
Batti "N" se vuoi uscire dal programma, "S" per ricominciare.
Applicati in questo gioco, allenati e diventerai il Dattilografo più veloce della Galassia!!

LABIRINTO

Nello strano mondo dei videogiochi tutto è possibile.
Infatti, in questo divertente confronto con il computer, tu sei un simpatico animaletto chiuso in un labirinto ed il tuo scopo è quello di raccogliere sacchi di denaro senza farti prendere dai Teschi, uno dei quali ti insegue.
I labirinti sono sempre diversi, come sempre diversa è la disposizione dei sacchi di denaro.
Devi studiare bene la strategia di gioco e con un po' di allenamento ti accorgerai che, in determinate posizioni, il Fantasma,

che segue a distanza le tue mosse, si blocca e così puoi ripulire interi settori senza essere inseguito.

I tasti controllo sono: E = alto
X = basso
D = destra
S = sinistra

Appena un Teschio ti mangia, per far riapparire il tuo animaletto nel labirinto, devi battere l'ultimo tasto di movimento premuto prima di essere ucciso.

Vai e arricchisciti!

HELP ME

Quale Cavaliere rimarrebbe insensibile alle grida di aiuto di una "Donzella"?

Allora parti "Lancia in resta" e vai a salvarla!

Un incantesimo, però, non rende facile la tua impresa: più passa il tempo e più delle barre di ferro ti impediscono il cammino.

I tuoi nemici sono:

A) Le barre di ferro: ti ostacolano, ma non ti uccidono

B) Il tempo: hai a disposizione un limitato numero di secondi, scaduti i quali sei un Uomo Morto. (Il contasecondi è la linea gialla in fondo allo schermo).

Le tue armi sono:

A) La tua spada "Excalibur" che ha il potere di aprirti un varco tra le barre di ferro

B) La velocità negli spostamenti.

I tasti controllo sono: E = alto
X = basso
D = destra
S = sinistra

Con il tasto "G" comandi la spada e ti apri un varco tutto intorno. Attento, però: hai a disposizione solo 20 colpi (ad ogni quadro finito, il computer sottrae quelli dati e aggiunge 4 colpi di bonus).

Vai, la Principessa ti aspetta!

Numeri già apparsi:

VIDEOTECA COMPUTER N. 1
per i possessori di Commodore 64
Nel manuale n. 1: I tasti funzionali del Commodore 64 - Pseudocodice e programmazione Basic - Il joystick.
Nella cassetta n. 1: Slalom - Slot Machine - Bilancio familiare - Briscola - Domino.

VIDEOTECA COMPUTER N. 2
per i possessori di Commodore 64
Nel manuale n. 2: Il basic più veloce - Una migliore gestione del video per il 64 - Disegnare con tastiera e joystick - Pseudocodice: 2ª lezione.
Nella cassetta n. 2: Tennis 3d - Totocalcio - Gestione magazzino - Wargame - Colour search.

VIDEOTECA COMPUTER N. 3
per i possessori di Commodore 64
Nel manuale n. 3: I cicli annidati del Basic - Come sviluppare un programma (Squash e Trampolino) - Conosci il tuo CBM 64?
Nella cassetta n. 3: Starway - Easyword - Poker - Forza 4 - Test Quoziente Intellettuale.

VIDEOTECA COMPUTER N. 4
per i possessori di Commodore 64
Nel manuale n. 4: Pseudocodice: Istruzioni read e data - Figura richiama - Grafico a barre. Il basic del CBM 64. I linguaggi di programmazione. Come personalizzare i programmi.
Nella cassetta n. 4: Dieta - Calorie dei cibi - Visischool - Sink - Hat in the ring.

VIDEOTECA COMPUTER N. 5
per i possessori di Commodore 64
Nel manuale n. 5: Pseudocodice: 5° lezione - I sistemi di numerazione - Le periferiche di ingresso e uscita. Basic: I cicli.
Nella cassetta n. 5: Esherland - Bosco incantato - Formula 1 - Coppa america - Geometria per le scuole medie.

PLAY ON TAPE N. 1
per i possessori di VIC 20
Nel manuale n. 1: I tasti funzionali del VIC 20 - Pseudocodice e programmazione Basic - Il joystick.
Nella cassetta n. 1: Totocalcio - Air attack - Cervellone - Inferno 3D - Bilancio familiare.

PLAY ON TAPE N. 2
Nel manuale n. 2: Il basic più veloce - I caratteri speciali del VIC 20 - Disegnare con la tastiera e il joystick - Pseudocodice: 2ª lezione.
Nella cassetta n. 2: Test per misurare il Quoziente intellettuale - Easyword - Caccia al tesoro - Gestione Magazzino - Formula 1.

PLAY ON TAPE N. 3
per i possessori di VIC 20
Nel manuale n. 3: I cicli annidati del Basic - Come sviluppare un programma (Corsa automobilistica - Piranha) - L'interfaccia sconosciuta - Conosci il tuo VIC 20?
Nella cassetta n. 3: Sette e mezzo - Dieta - Calorie dei cibi - Gangster - Costi chilometrici.

PLAY ON TAPE N. 4
Per i possessori di VIC 20
Nel manuale n. 4: Pseudocodice: Istruzioni read e data - Figura richiama - Grafici a barra. Il basic del VIC 20 - I linguaggi di programmazione - Come personalizzare i programmi.
Nella cassetta n. 4: G.O MO-KU - Calcolatrice - Golf - Vic Tab - Cabala e sogni.

COMPUTING VIDEOTECA N. 1
Per i possessori di Sinclair
SPECTRUM ZX
Nel manuale n. 1: Pseudocodice e programmazione basic: 1ª e 2ª lezione - La gestione dei canali dello Spectrum - Come farsi una cassetta di "Subroutines".
Nella cassetta: Wargame - Gestione del magazzino - U.F.O. - Helibomber.

COMPUTING VIDEOTECA N. 2
per i possessori di Sinclair
ZX SPECTRUM
Nel manuale n. 2: I cicli annidati del Basic - Come sviluppare un programma (Gara di sci) - Variabili interne e gestione del video nello Spectrum.
Nella cassetta n. 2: Frog Race - Pac Chian - Torre Laser - Dieta - Calorie dei cibi.

COMPUTING VIDEOTECA N. 3
per i possessori di ZX SPECTRUM e
TI-99/4A
Nel manuale n. 3: Pseudocodice: 4ª lezione - Il Basic dello Spectrum - I linguaggi di programmazione - Ti Super Sound.
Nella cassetta n. 3. Lato ZX Spectrum: Buio (48 K) - Anatomia (48 K) - Brain (48 K) - Visischool (16 K) - Escape (16 K).
Lato TI-99/4A: Poker - Planets explorer - Char designer - Forza 4 - Plat (extended Basic).

COMPUTING VIDEOTECA N. 4
per i possessori di ZX SPECTRUM e
TI-99/ 4A
Nel manuale n. 4: Pseudocodice: strutture dati - Coda lista - Simulazione. I sistemi di numerazione. Le periferiche di ingresso e uscita. Il Basic: 2° lezione.
Nella cassetta n. 3. Lato ZX Spectrum: Mixxil - Dizionario di inglese - Sequencer - Lines - Bioritmo.
Lato TI-99/4A: Space War - Slot machine - Agente segreto - Dieta - Calorie.

**I numeri arretrati costano L. 15.000. Indirizzare vaglia o assegno a Editoriale VIDEO via Castelvetro 9 - 20154 Milano specificando il numero richiesto.
Ufficio tecnico e arretrati: telefono 02/3184829**

COMPUTING
VIDEOTECA
5
ZX SPECTRUM
• PING PONG
• SQUARE
• CAVERN
• 3D GRAFF
• SPACE TRAFIC
TI 99/4A
• CASSAFORTE
• MEDIA-VOTI
• LETTERE ASSASSINE
• LABIRINTO
• HELP ME
EV EDITORIALE VIDEO